



REMARKS

Applicants respectfully traverse and request reconsideration.

Claims 1 through 4, 6, 10 through 13 and 15 stand rejected under 35 U.S.C. § 102(a) as being anticipated by U.S. Patent No. 6,173,394 (Guttag et al.). The Office Action cites col. 130, line 42 through col. 131, line 51, for allegedly teaching all of the claim limitations of these claims.

Guttag is directed to a data processing apparatus where a memory store operation comes from one of a pair of registers selected by an arithmetic logic unit condition. A register pair conditional store instruction designates particular ones of the status bits protected from being set by the ALU output. The register pair conditional store instruction designates a particular one of the status bits to control whether data as stored in the first or second register is stored in the specified memory address. A status bit protect instruction permits selection of status bits protected from modification corresponding to the current arithmetic logic unit result.

In contrast, the amended claims inquire, among other things, emulating non-native instructions using native instructions that contain a flag modification enable bit and determining whether the flag modification enable bit allows updating of at least one flag in response to executing the operational code. The Guttag reference is silent as to any such step or operation. For these and other reasons, the pending claims are not anticipated.

Claims 5, 7 through 9, 14 and 16 through 18 stand rejected under 35 U.S.C. § 103(a) as being obvious in view of Guttag and applicants specification. Applicants present originally filed claims 7 and 16 for consideration and as such first address the rejection in view of these claims. The Office Action alleges that one of ordinary skill in the art would have been motivated to utilize Guttag's teaching of bits within the instruction controlling modification of condition codes with a prior art native and non-native simulation system because such a combination would allegedly enable finite calm instruction level control of such modification without the necessity of storing values and all the related control issues as in prior art systems. This appears to be conclusory. There does not appear to be factual support for a motivation to combine selected teachings. For example, Guttag is silent as to any type of emulation scheme and appears to be directed to a completely different problem. Moreover there is no disclosure in Guttag or applicants admitted prior art regarding emulating non-native instructions using native instructions containing the flag modification enable bit. Applicants respectfully submit that any

motivation must come without the benefit of the applicants' invention or knowledge of the applicants' invention. It appears that impermissible hindsight is being used.

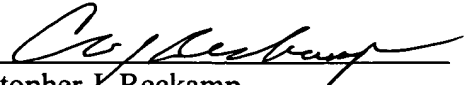
In addition, the Gutttag reference appears to teach that the conditional operations permit selection of different registers; for example, using the same instruction set (see for example 156 lines 41 through 50). There is no emulation of non-native instructions using native instructions that contain flag modification enable bits.

In addition, claim 5 requires, for example, providing a variable length instruction emulator that uses fixed length native instructions to emulate variable link instructions in evaluating the flag modification enabled bit to preserve flag bit settings for variable length instructions that are emulated using the fixed blank native instructions. Applicants again reassert the relevant remarks made with respect to the failure to show adequate motivation and also note that the Gutttag reference is not directed to a variable length instruction emulation circuit or method. Again it appears that the motivation comes from applicants' own invention.

With respect to claims 9 and 18, applicants note that these claims require, among other things, converting received variable length X86 instructions to a plurality of native instructions when the native instructions have a flag modification enable bit set to allow changing of the non-native instruction flags in response to execution of the native instructions. Again, Gutttag is silent as to any instruction conversion. Moreover, the claim requires, that as to unconvertible instructions, the non-native instruction emulator emulates unconverted variable length instructions that include flag modification enable bits set to prevent changing of non-native instruction flags in response to the execution of the native instructions for the unconverted variable length instruction. A setting of the modification enable bits based on conversion and unconvertible instructions is not taught or suggested by the combination of teachings. Accordingly these claims are also believed to be in condition for allowance.

Applicant respectfully submits that the claims are in condition for allowance and respectfully request that a timely Notice of Allowance be issued in this case. The Examiner is invited to contact the below listed attorney if the Examiner believes that a telephone conference will advance the prosecution of this application.

Respectfully submitted,

By: 
Christopher J. Reckamp
Reg. No. 34,414

Dated: October 1, 2002

Vedder, Price, Kaufman & Kammholz
222 North LaSalle Street
Chicago, Illinois 60601
Telephone: (312) 609-7599
Facsimile: (312) 609-5005

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Claims:

1. (Amended) A method for processing program instructions comprising the steps of:

receiving at least one instruction containing data representing operational code and data representing at least one flag modification enable bit;

emulating non-native instructions using native instructions containing the flag modification enable bit.

determining whether the at least one flag modification enable bit allows updating of at least one flag in response to executing the operational code; and

updating at least one flag in response to determining a status of the at least one flag modification enable bit.

8. (Amended) The method of claim 1[7] wherein the non-native instructions are variable length X86 instructions.

10. (Amended) An apparatus for processing program instructions comprising:

a buffer coupled to receive at least one instruction containing data representing operational code and data representing at least one flag modification enable bit; [and]

a variable length instruction emulator that uses fixed length native instructions as the at least one instruction, to emulate variable length instructions wherein the variable length instruction emulator emulates non-native instructions using native instructions containing the flag modification enable bit; and

a controller operatively responsive to the at least one instruction stored in the buffer, that determines whether the at least one flag modification enable bit allows updating of at least one flag in response to executing the operational code and that updates at least one flag in response to determining a status of the at least one flag modification enable bit.

17. (Amended) The apparatus of claim 10[16] wherein the non-native instructions are variable length X86 instructions.